

# Cryptographic Protocols

## Exercise 11

### 11.1 Consensus: An Example

Four players  $P_1, \dots, P_4$  execute the **Consensus** protocol from the lecture with  $t = 1$ , where  $P_1$  is corrupted and has the following strategy: in any step of the protocol where the players are instructed to send a value to all other players,  $P_1$  always sends the value 1 to  $P_2$  and  $P_3$ , and the value 0 to  $P_4$ . Below, you can find a table with the outputs of the players in the sub-protocols in an execution of the **Consensus** protocol in the above setting.

	$P_1$	$P_2$	$P_3$	$P_4$
Input	–	1	0	0
WeakConsensus	–	⊥	⊥	0
GradedConsensus	–	(0, 0)	(0, 0)	(0, 0)
KingConsensus $_{P_1}$	–	1	1	0
WeakConsensus	–	1	1	⊥
GradedConsensus	–	(1, 1)	(1, 1)	(1, 0)
KingConsensus $_{P_2}$	–	1	1	1

a) Fill, similarly to the above example, the following tables (where the strategy of  $P_1$  is as described above):

**Scenario 1:**

	$P_1$	$P_2$	$P_3$	$P_4$
Input	–	1	1	0
WeakConsensus	–			
GradedConsensus	–			
KingConsensus $_{P_1}$	–			
WeakConsensus	–			
GradedConsensus	–			
KingConsensus $_{P_2}$	–			

**Scenario 2:**

	$P_1$	$P_2$	$P_3$	$P_4$
Input	–	1	1	1
WeakConsensus	–			
GradedConsensus	–			
KingConsensus $_{P_1}$	–			
WeakConsensus	–			
GradedConsensus	–			
KingConsensus $_{P_2}$	–			

b) Is it possible in the scenarios from a) that the honest players decide on 0? Describe the corresponding strategy for  $P_1$  or justify why such a strategy does not exist.

c) Assume that  $P_1, P_2$ , and  $P_3$  all have input 1 and  $P_4$  has input some  $b \in \{0, 1\}$ . Show (without computing the corresponding table) that when at most one player is corrupted, then the construction

$$(\text{WeakConsensus}; \text{GradedConsensus};) \text{KingConsensus}_{P_4}$$

achieves the properties of Consensus.

## 11.2 Variations of GradedConsensus

- a) Amélie has an idea for improving the `GradedConsensus` protocol from the lecture: in step 3, each player  $P_j$  computes the value  $y_j$  as follows:

$$y_j = \begin{cases} 0 & \text{if } \#\text{zeros} \geq n - t, \\ 1 & \text{otherwise.} \end{cases}$$

What do you think about this suggestion? Does the new protocol achieve GRADED CONSENSUS? Justify your answer.

- b) Cindy also has a suggestion, where  $y_j$  is computed as follows:

$$y_j = \begin{cases} 0 & \text{if } \#\text{zeros} > t, \\ 1 & \text{if } \#\text{ones} > t, \\ \text{random} & \text{otherwise.} \end{cases}$$

What do you think about Cindy's suggestion? Is the new protocol well-defined? Does it achieve GRADED CONSENSUS? Justify your answer.

- c) Hans has yet another idea: compute  $y_j$  as in the protocol from the lecture, but change the way  $g_j$  is computed so that it is set to 1 when more than half of the received values equal  $y_j$ , i.e.,

$$g_j = \begin{cases} 1 & \text{if } \#y_j\text{'s} > n/2, \\ 0 & \text{otherwise.} \end{cases}$$

Does Hans' protocol achieve GRADED CONSENSUS? Justify your answer.

## 11.3 Broadcast of Long Messages

In the lecture, we have seen a broadcast protocol for bits. One can construct a protocol for  $\ell$ -bit strings  $x \in \{0, 1\}^\ell$  by invoking the binary broadcast protocol  $\ell$  times (in parallel). However, for large  $\ell$ , this is very inefficient.

The following protocol achieves broadcast for  $\ell$ -bit messages by invoking bit-broadcast only  $n$  times:

1. The sender  $P_s$  sends  $x$  to each  $P_i$ . We denote the message received by  $P_i$  as  $x_i$ .
  2. Each  $P_i$  sends  $x_i$  to every  $P_j$ . We denote the message received by  $P_j$  as  $x_{ij}$ .
  3. For each  $P_i$ , if  $P_i$  received  $x_{ij} = x_i$  from at least  $n - t$  parties  $P_j$  (formally, i.e.  $|\{j : x_{ij} = x_i\}| \geq n - t$ ), broadcast a 1-bit. Otherwise, broadcast a 0-bit.
  4. Let  $\mathcal{M}$  be the parties that broadcast 1. If  $|\mathcal{M}| < n - t$ , then every  $P_i$  outputs  $\perp$ . Otherwise,  $P_i$  outputs  $y_i$ , where  $y_i$  is the value  $P_i$  received most often from parties in  $\mathcal{M}$ .
- a) Show that if the sender is honest, the value the honest players output is the value sent by him, that is,  $y_i = x$  for every honest  $P_i$ .
- b) Show that  $x_i = x_j$  for any two honest parties  $P_i, P_j \in \mathcal{M}$ .
- c) Show that all honest players output the same value. That is,  $y_i = y_j$  for any two honest parties  $P_i, P_j$ .